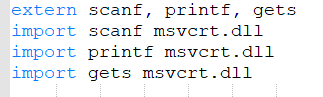
**Seminar - Reading from keyboard and Printing on screen in Assembly**

**CDECL convention rules:**

* Arguments of an external function are placed on the stack from right to left (**an element from a stack is a doubleword**);
* The result returned by the external functions are stored in EAX register;
* Registers EAX, ECX, EDX are used within the functions (so they can be overwritten) so if we need the values stored in EAX, ECX, EDX we need to store them (in auxiliary variables for instances or using PUSHAD before using the external functions and POPAD after we execute the external functions) before the function call.
* The external functions don’t empty the stack; it is the responsibility of the programmer to pop the arguments out of the stack after the function call (add esp, 4\*number of doublewords saved on the stack)

**The external functions** used are scanf, printf and gets.

Before using these functions, the functions are **declared and imported:**



**Syntax:**

int scanf (const char \* format, variable\_address\_1, ...);

int printf (const char \* format, variable\_1, constant\_2, ...);

|  |  |  |  |
| --- | --- | --- | --- |
| Code  FORMAT | Type | Example | Value representation dimension |
| c | Character | a | byte |
| **d** | Signed decimal integer | 392 | dword |
| u | Unsigned decimal integer | 7235 | dword |
| **x** | Hexadecimal integer | 7fa | dword |
| **s** | String (terminated with a 0 - asciiz) | example | string of bytes terminated with 0 |

The strings represented the format have to be in ASCIIZ format (last element from the string=0)

**Example: read a number in decimal and print the number in hexadecimal**

|  |  |
| --- | --- |
| **Data segment**  readformat db '%d', 0  printformat db '%x', 0  nr dd 0 | **Code segment**  ;citire in zecimal  push dword nr  push dword readformat  call [scanf]  add esp, 4\*2  ;afisare in hexazecimal  push dword [nr]  push dword printformat  call [printf]  add esp, 4\*2 |

To execute the code we select **RUN Program** from ASM Plugins.

We cannot use in assembly \n (new line) as in others languages, therefore we need to find a solution: the ascii code for newline:



ex:

|  |  |
| --- | --- |
| data segment  newline db 10, 0 | code segment  push dword newline  call [printf]  add esp, 4\*1 |

Functia gets

gets (string) – allow us to read a string from the keyboard with multiple spaces:

Example: Read a string and print the string on the screen:

|  |  |
| --- | --- |
| **Data segment**    s res**b** 20  format\_sir db '%s'**,** 0  mesaj db 'S-a citit acest sir:'**,** 0 | **Code segment**  **push** **dword** s  **call** **[**gets**]**  **add** **esp,** 4**\***1    ;print pe ecran mesajul    **push** **dword** mesaj  **call** **[**printf**]**  **add** **esp,** 4**\***1    ; print pe ecran sirul citit    **push** **dword** s  **push** **dword** format\_sir  **call** **[**printf**]**  **add** **esp,** 4**\***2 |

**Problems:**